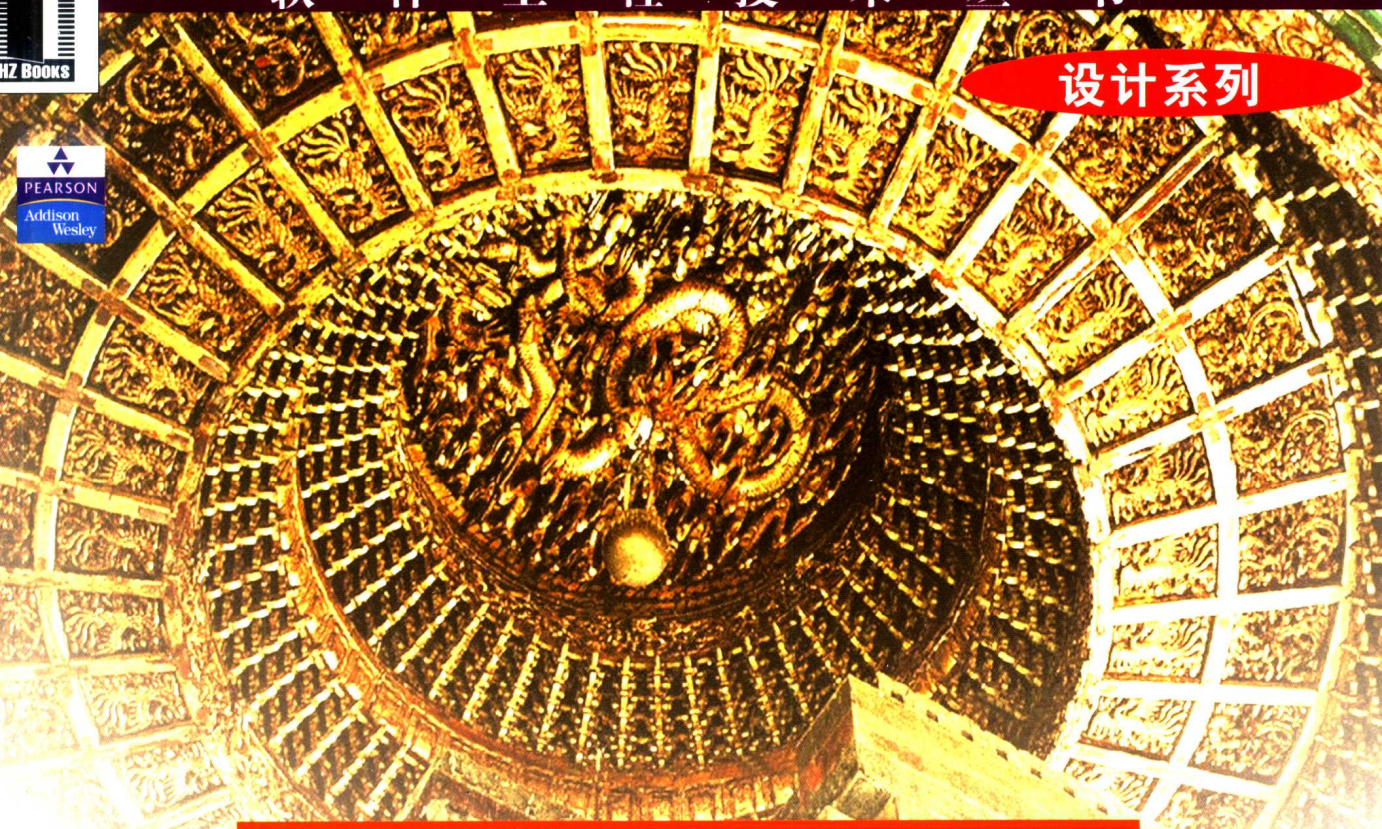
软 件 工 程 技 术 丛 书

设计系列



# 企业应用架构模式

## Patterns of Enterprise Application Architecture

（英） Martin Fowler 著
王怀民 周斌 译
UMLChina 审校

software development productivity award

机械工业出版社
China Machine Press

Patterns of Enterprise Application Architecture

# 企业应用架构模式

**企业应用开发的实践得益于多种新技术的出现。多层的面向对象平台（如Java、.NET）已经日渐平常。这些新工具和新技术有能力构建更强大的企业应用程序，但是在实现上还不太容易。由于开发人员未能充分理解有经验的对象程序开发人员在架构方面的经验和教训，因此企业应用中经常存在一些共同的错误。**

本书就是面向企业应用开发者的，可帮助他们迎接这种艰难挑战。本书的作者Martin Fowler注意到，尽管技术本身存在变化——从Smalltalk到CORBA，再到Java和.NET，但基本的设计思想并没有太多变化，可以加以适当调整，用来解决那些共同的问题。在一组专家级合作者的帮助下，作者将40多种经常出现的解决方案转化成模式，最终写成这本能够应用于任何一种企业应用平台的、关于解决方案的、不可或缺的手册。本书曾于2002年荣获美国“Software Development”杂志的图书：通用类生产效率奖和读者选择奖。

本书涉及两部分内容。第一部分是关于如何开发企业应用的简单介绍。在阅读这部分时，读者可以从头到尾通读，以掌握本书的范围。第二部分是本书的主体，是关于模式的详细参考手册，每个模式都给出使用方法和实现信息，并配有详细的Java代码或C#代码的示例。此外，整本书中还用了大量UML图来进一步阐明有关概念。

## 本书涵盖如下方面：

- 将企业应用分层
- 组织企业业务逻辑的主要方法
- 在对象和关系数据库之间进行映射的深层次解决方案
- 通过模型-视图-控制器来组织Web表现
- 处理跨多事务的数据的并发问题
- 设计分布式对象接口

## 作者介绍：

**Martin Fowler** 在面向对象分析设计、UML、模式、软件开发方法学、XP、重构等方面，都是世界顶级的专家，现为ThoughtWorks公司的首席科学家。ThoughtWorks是一家从事企业应用开发和集成的公司。早在20世纪80年代，Fowler就是使用对象技术构建多层企业应用的倡导者，他著有几本经典书籍：《分析模式》、《UML精粹》和《重构》等。

封面制作 锡彬

ISBN 7-111-14305-1

9 787111 143055

华章图书

华章网站 http://www.hzbook.com

网上购书：www.china-pub.com

北京市西城区百万庄南街1号 100037

读者服务热线：(010)68995259，68995264

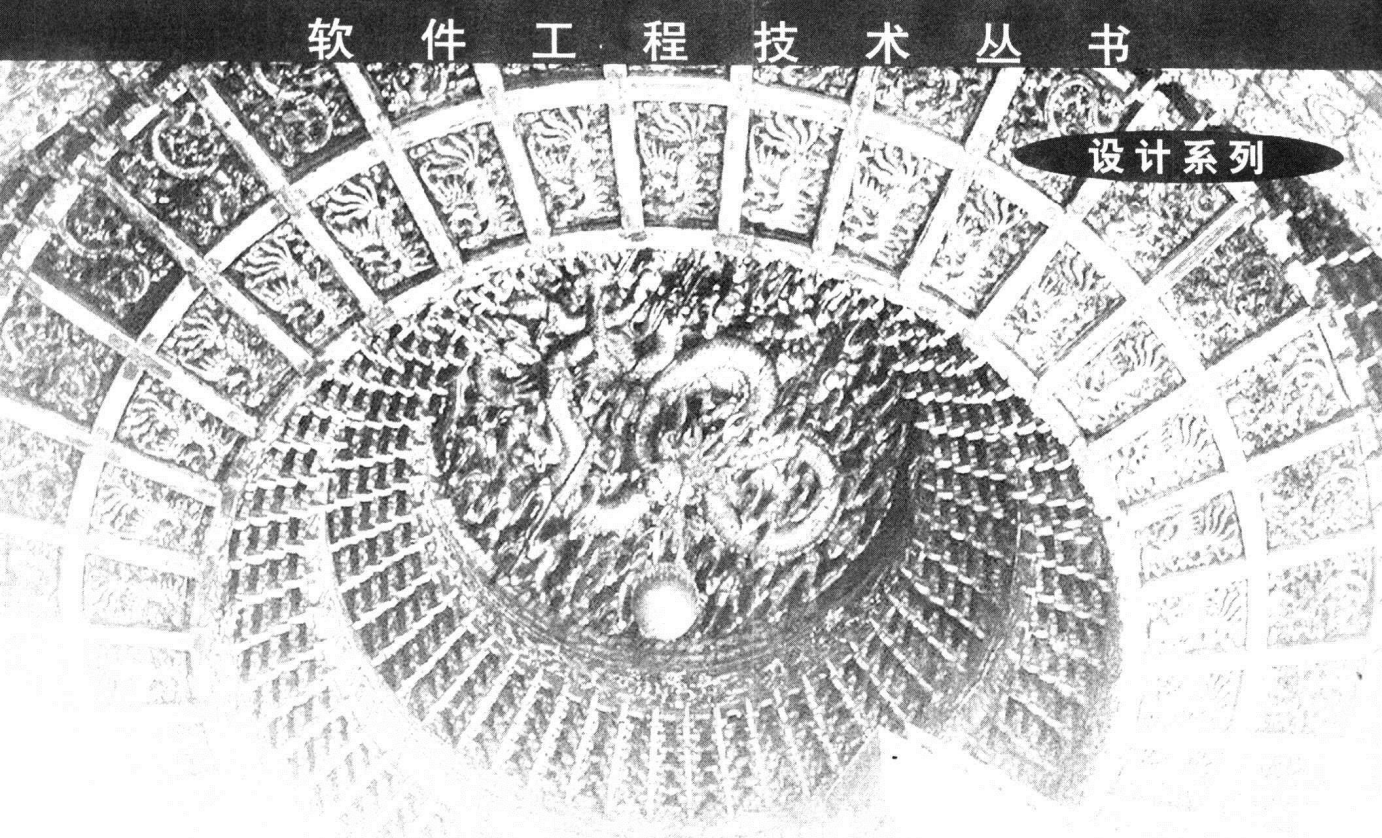
读者服务信箱：hzedu@hzbook.com

ISBN 7-111-14305-1/TP · 3544

定价：49.00 元

软件工程技术丛书

设计系列



# 企业应用架构模式

Patterns of Enterprise Application Architecture

（英） Martin Fowler 著

王怀民 周斌 译

UMLChina 审校

机械工业出版社
China Machine Press

本书作者是当今面向对象软件开发的权威，他在一组专家级合作者的帮助下，将40多种经常出现的解决方案转化成模式，最终写成这本能够应用于任何一种企业应用平台的、关于解决方案的、不可或缺的手册。本书获得了2003年度美国软件开发杂志图书类的生产效率奖和读者选择奖。本书分为两大部分。第一部分是关于如何开发企业应用的简单介绍。第二部分是本书的主体，是关于模式的详细参考手册，每个模式都给出使用方法和实现信息，并配以详细的Java代码或C#代码示例。此外，整本书中还用了大量UML图来进一步阐明有关概念。

本书是为致力于设计和构建企业应用的软件架构师、设计人员和编程人员而写的，同时也可作为高等院校计算机专业及软件学院相关课程的参考教材。



**本书版权登记号：图字：01-2003-0997**

**图书在版编目（CIP）数据**

企业应用架构模式 /（英）福勒（Fowler, M.）著；王怀民等译. - 北京：机械工业出版社，2004.7

（软件工程技术丛书 设计系列）

书名原文：Patterns of Enterprise Application Architecture

ISBN 7-111-14305-1

Ⅰ.企… Ⅱ.①福… ②王… Ⅲ.软件工具-程序设计-应用-企业管理 Ⅳ.F270.7

中国版本图书馆CIP数据核字（2004）第029141号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：姚 蕾

北京昌平奔腾印刷厂印刷 · 新华书店北京发行所发行

2004年7月第1版第1次印刷

787mm × 1092mm 1/16 · 24印张

印数：0 001-5 000册

定价：49.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

# 译者序

“每一个模式描述了一个在我们周围不断重复发生的问题，以及该问题的解决方案的核心。这样，你就能一次又一次地使用该方案而不必做重复劳动。”

——*Christopher Alexander*

本书是面向对象大师Martin Fowler继《Analysis Patterns》、《UML Distilled》、《Planning Extreme Programming》、《Refactoring》之后的又一力作。

“温故而知新”。Fowler在本书中再次向我们证明了《礼记》中这句古训的震撼力——他在回头审视自己及同仁多年来从事企业应用开发的经验和教训后，归纳总结了40多种企业应用架构的设计模式。这些模式从不同层次、不同侧面向我们展示了什么是好的企业应用架构？如何设计好的企业应用？

正如作者自己所言，企业应用在某些方面比其他软件（如电信通信软件）复杂得多：纷繁复杂的企业数据、“不合逻辑”的业务规则、变化莫测的用户需求，等等。环顾四周——CORBA、J2EE、.NET——企业应用开发技术可谓“前仆后继、层出不穷”，开发平台的种类之多就更不必说。

招式套路可以千变万化，扎实深厚的“内功”却是始终如一！虽然企业应用涉及的软件技术不断翻新，但是基本的架构及设计思想却没有太多变化。将以前行之有效的设计思路和方法加以适当调整，并应用到当前的问题上，是最高效的做法。在一组专家级合作者的帮助下，Martin将40多种经常出现的解决方案转化成模式，最终融会成这本“内功心法”。在仔细研读、用心揣摩本书之后，希望它能够帮助你应对任何一种企业应用平台，驾御任何一种企业应用技术——无论是现在的技术还是未来的技术。

熟悉Fowler的读者都知道，这位大师的写作风格可谓是“深入浅出，娓娓道来”。本书也是一样。前8章是关于企业应用的背景知识，如分层架构、Web表现、业务逻辑、数据库映射、并发、会话、分布策略，等等。在此基础上，随后的各章分别对与这些背景知识相关的设计模式进行了详细的介绍。与其他设计模式的书一样，本书从模式的使用场景、解决方案、UML表示等方面予以介绍，详略有致。就连示例的编程语言的选取——Java和C#——也是与他的写作风格一脉相承的。

夜已深，窗外依旧是绵绵不断的早春小雨。让我们酌一杯清茶，一起来品味大师的话，一起来品味“源于实践、指导实践”的苦涩与甘甜——

“模式的关键点是它们源于实践。必须观察人们的工作过程，发现其中好的设计，并找出‘这些解决方案的核心’。这不是一个简单的过程，但是一旦发现了某个模式，它将是非常有价值的。对于我来说，价值之一是能够撰写这样一本参考书。你不必通读本书的全部内容，也不必通读任何一本有关模式的书。只需要了解到这些模式都是干什么的、它们解决什么问题、它们是如何解决问题的，就足够了。这样，一旦你碰到类似问题，就可以从书中找出相应的模式。

那时，你再深入了解相应的模式也为时不晚。”

不闻不若闻之
闻之不若见之
见之不若知之
知之不若行之
——荀子

本书翻译初稿的过程中得到了丁博、王树凤、朱锐、林繁、托明福的大力帮助。王怀民、周斌分别统审了全书。非常感谢UMLChina潘加宇、蒋芳在校对过程中的宝贵意见和建议。

译 者

2004年3月27日于长沙

# 前　言

1999年春天，我飞抵芝加哥为ThoughtWorks公司正在开发的一个项目担任顾问。ThoughtWorks是一个规模虽小但正在快速成长的应用开发公司。这个项目属于那种极富挑战性的企业级应用，它是一个后端租赁系统。简单说，它处理的是租户签字认可后所有与租赁有关的事务，包括发送账单、处理某些人对所租房屋资产的改造、追踪那些未按时缴纳账单的租户、指出如果某人提前归还资产应当如何处理等。在你意识到租赁合同极度复杂并且总在不断变化之前，这听起来好像并不太难实现。它的业务“逻辑”几乎不能套用任何已有的逻辑模式，因为那些“逻辑”归根到底是商人们为争夺生意而制定的，一些古怪的小改动都可能对赢得某笔交易起关键作用。因此，每次生意上的一点点胜利就意味着系统复杂性的又一次增加。

我对此类问题情有独钟：如何捕获这些复杂性，并设计一个面向对象的系统来处理它们。事实上，我一直坚信面向对象的最大优点在于它能够使复杂逻辑易于处理。为复杂业务逻辑开发一个良好的领域模型很困难，但在此问题中却是恰得其所。

当然，单纯的领域模型也并非灵丹妙药。我们的领域模型必须持久化到数据库中，但是与许多项目一样，我们当时使用的是关系数据库。我们还必须将该模型与用户界面关联起来，还要能支持远程应用程序对本软件的使用，同时还必须将我们的软件与第三方的软件包整合。这些工作都基于一种被称为J2EE的新技术，当时全世界没有人在这一方面有实战经验。

虽然J2EE是一项崭新的技术，但我们仍能从以往的经验中得到帮助。本人曾经长期使用C++、Smalltalk和CORBA来开发类似系统。ThoughtWorkers公司中许多人则有Forte方面的经验。可以说当时我们对关键架构已经有了思路，所要解决的仅是如何将之转化为J2EE实现。现在回过头来看三年前的设计，虽然它并不尽善尽美，但无疑经受住了时间的考验。

上述情况正是我撰写这本书的出发点。多年来，我曾看到过许多企业级应用项目。这些项目通常都包含相似的设计思路，这些设计思路已经被证明可以有效地处理企业应用中不可避免的复杂性。本书就是将这些设计思路升华为模式的一个起点。

本书分为两个部分，第一部分是一些叙述性的章节，它们主要讨论企业级应用程序设计中的一些重要议题。这些章节介绍了企业级应用程序架构的各种问题，并给出了大体的解决方案。而解决方案的细节则在本书的第二部分以模式的方式组织成文。这些模式仅仅是一些参考，我并不希望读者一页页地去细读。我的想法是：从头到尾将第一部分的叙述性章节读完，然后再根据兴趣和需求翻阅第二部分的有关章节。因此，本书是将简短的叙述和详尽的参考合二为一。

本书讨论的是企业级应用程序的设计。企业级应用程序涉及到大量复杂数据的显示、操纵和存储以及对这些数据进行处理的业务流程的自动化。典型的例子有预订系统、金融系统、物流补给系统以及其他种种驱动现代商业运作的系统。企业级应用和嵌入式系统、控制系统、电信系统或者桌面应用程序不同，它们有自己特有的挑战和解决方案。因此，如果你在上述那些非企业领域工作，本书对你并没有益处（除非想体会一下企业级应用程序是怎么一回事）。如果

需要一本关于软件架构的通用性的书籍，我推荐[POSA][⊖]。

在构建企业级应用时有许多架构方面的问题。我想本书恐怕很难一一详细列举。在软件开发方面，本人是迭代开发方法的忠实信徒。迭代开发的核心在于只要软件对用户有用，就应当交付，即使这个软件当时并没有完成。虽然著书与编写软件之间存在诸多差异，但我认为在这一点上异曲同工。也就是说，虽然本书尚不全面，但已初具雏形，可以为读者提供有关企业级应用程序架构方面的有益建议（至少我认为如此）。本书讨论的主要议题是：

- 企业级应用程序的分层
- 构建领域（业务）逻辑
- 构建基于Web的用户界面
- 将内存模块（尤其是对象）关联到关系数据库
- 在无状态环境下处理会话状态
- 分布原则

可能本书未涉及到的议题更多。我很想撰写关于组织确认、合并消息和异步通信、安全、错误处理、集群、应用集成、架构重构、构建胖客户的用户界面等方面的书籍或文章。但是，由于时空限制以及思路尚未成熟，本书将不涉及上述内容。我只能希望在不久的将来能看到一些与这些工作相关的模式。也许我会撰写本书的第2卷并加入这些内容，或者是由其他人来补遗。

当然，基于消息的通信是相当重要的问题。在进行多应用程序集成时，人们正越来越多地用到异步的、基于消息的通信方法。即便是在同一应用程序内部，基于消息的通信也值得费上一些笔墨。

本书并非针对某一特定的软件平台。从20世纪80年代末到90年代初，我开始在基于Smalltalk、C++和CORBA的项目中使用这些模式。而20世纪90年代后期我在Java方面做了大量的工作，我发现这些模式可以很好地应用于较早的Java/CORBA系统和其后基于J2EE的工作中。近来，我开始在微软的.NET平台方面做一些探索，我发现这些模式同样有效。ThoughtWorkers公司的同事也介绍了他们的经验，尤其是在Forte方面。我不敢说这些模式能够通用于所有已经和即将被用于企业级应用的开发平台，但至少到目前为止，它们已经表现出足够的可重用性。

对于大多数模式，本书提供了相应的代码示例。这些例子所用的程序设计语言是我认为大多数读者都能够阅读和理解的语言。Java就是一个很好的选择。只要熟悉C或C++就可以读懂Java代码，Java远没有C++那么复杂。基本上，大多数C++程序员都能够理解Java，但反过来却并非如此。我是面向对象的信徒，也就自然更偏爱面向对象的语言。因此，大多数代码示例使用的是Java语言。写这本书时，微软的.NET环境正逐渐成熟，它的C#语言与Java有许多相同之处。所以某些代码示例也使用了C#语言，虽然这样做会多少有一些风险，因为.NET尚未得到大量应用，使用的术语可能尚未形成惯例。这两种语言都是基于C的语言，只要能读懂其中一种，即使对另一种语言或平台并不熟悉，要读懂它也并非难事。我的目的是使用一种能够让最多的软件开发者读懂的语言，即使这种语言并非他们所擅长或偏爱的。（谨向那些喜欢Smalltalk、Delphi、Visual Basic、Perl、Python、Ruby、COBOL或其他语言的读者致歉。我知道你们认为

---

⊖ 本书的中文版（中译名为“面向模式的软件体系结构”）已由机械工业出版社出版。——编辑注

有比Java或C#更好的语言，我也认为如此！）

示例是用来阐述和解释模式思想的。它们并非可以直接使用的解决方案；任何情况下都需要做一些工作才能将它们用于你的应用程序之中。模式只是一个有益的起点，而非最终的解决之道。

## 本书的读者

本书面向的是正在构建企业级应用、希望增进对架构相关问题的理解和沟通的编程人员、设计人员和软件架构师。

我假定本书的大多数读者可以归为两类：一些人所面对的需求并非大规模的，因此准备自己从零开始构建软件；另一些人则有大规模的需求，将使用某些工具。对于前者，我认为本书中提及的模式将有助于工作的启动。尽管在许多领域，所需的知识远超出本书中模式所给予的内容，但本书提供了一个比我当初进入该领域时更高的起点。对于那些工具的使用者，我希望本书能揭开工具的一些内幕，帮助他们决策如何选择工具所支持的模式中的哪一种。例如，使用对象－关系映射工具同时意味着你必须决定如何映射某些特定情况。本书所提供的模式将有助于你做出这样的决策。

当然，还可能有第三类读者，他们有大规模的需求，同时又希望亲自构建软件。在此，我首先要忠告他们的是：请先考虑使用已有的工具。我已经见过不止一个项目花费了大量时间来建造框架，而这些框架并非该项目所真正要解决的问题。如果你仍固执己见，那也只能听天由命。但要记住，本书中代码示例为提高可理解性都被有意简化过，在实际使用时，往往需要对它们大动干戈才能满足要求。

由于模式是可复现问题的通用解决方案，因此可能有的读者对这些模式已经有所接触。如果你从事企业级应用开发已经有一段时间，可能会很熟悉其中大部分模式。本书中并不包含任何新的东西，正相反：这是一本关于（我们这一行业的）已有知识的书。如果你是这一领域的新手，我希望本书将帮助你学习这些技术。如果你熟悉这些技术，我希望本书有助于你与其他人沟通。模式的重要作用就在于其创建了一个通用的词汇表，例如，你称某个类是远程外观，其他设计人员就都知道你指的是什么。

## 致谢

与其他书籍一样，本书将涉及到多年来以不同方式与我一起工作的很多人。他们以许多方式对本书提供过帮助。本书中一些重要内容是由他人提供的，其中某些人我可能已经无法回忆起他们的名字。在这里，我所能做的是对那些仍铭记在心的人表示感谢。

首先要感谢的是我的合作者，David Rice,他是我在ThoughtWorks公司的同事，他为本书做出了巨大的贡献：撰写了本书的1/10。当我们一起努力以保证本书能如期交付时（他当时还要从事客户支持的工作），我们曾在若干个深夜通过即时消息进行协商，交谈中他坦承他总算明白了为何写一本书是如此困难却又如此吸引人。

Matt Foemmel是ThoughtWorks公司的另一个同事。尽管他文笔犀利冷峻，是本书的一个十分中肯的批评家，但他为代码示例做出了很大的贡献。Randy Stafford为本书贡献了服务层模式，

他一直是此模式的极力倡导者。我还要感谢Edward Hieatt和Rob Mee所做的贡献，特别是Rob在复审本书时所发现的缺失。Rob是我的最佳审阅者：他不仅发现少了某些内容，而且他还帮我写了一节来弥补这个缺失！

同样，对本书一流的正式审阅者，我的言辞远不能表达我的感激之情：

| | |
|---|---|
| John Brewer | Rob Mee |
| Kyle Brown | Gerard Meszarios |
| Jens Coldewey | Dirk Riehle |
| John Crupi | Randy Stafford |
| Leonard Fenster | David Siegel |
| Alan Knight | Kai Yu |

我几乎要把ThoughtWorks公司的电话号码簿列在此处了，因为太多同事与我讨论过他们的设计和经验，在这一项目上帮助过我。许多模式在我脑中成型，是因为我有机会与众多天才设计师讨论，因此我只好对整个公司表示感谢。

Kyle Brown、Rachel Reinitz和Bobby Woolf在百忙之中抽出时间与我一道在北卡罗莱纳对本书进行了长期而细致的审阅。他们在本书中注入了他们睿智的光芒。尤其是与Kyle的几次长时间的电话交谈令我获益匪浅。

2000年初我与Alan Knight和Kai Yu一起为Java One大会准备了一个演讲，这是本书最初的雏形。在对他们所提供的帮助致谢的同时，我还要感谢Josh Mackenzie、Rebecca Parsons和Dave Rice，他们其后协助我提炼了这些演讲及其思想。Jim Newkrik付出了很大努力协助我熟悉.NET平台。

我与这个领域的许多专家有过令人惬意的交谈或合作，从而在他们身上学到了不少东西。尤其想对Colleen Roe、David Muirhead和Randy Stafford表示感谢，他们将自己在Gemstone的Foodsmart示例系统上的工作成果与我共享。我在Bruce Eckel所主持的Crested Butte讨论会上也参与过一些重要的会谈，因此应当向近年来这一会议的与会者致谢。Joshua Kerievsky虽然没有时间对本书做一次全面的审阅，但他是模式方面的一个优秀顾问。

我还从UIUC阅读组那里获得了相当大的帮助，他们作为读者对本书提出了坦诚的批评。我要感谢：Ariel Gertzenstein、Bosko Zivaljevic、Brad Jones、Brian Foote、Brian Marick、Federico Balaguer、Joseph Yoder、John Brant、Mike Hewner、Ralph Johnson和 Weerasak Witthawaskul。

前UIUC成员Dragos Manolescu及其小组给了我一些反馈。感谢Muhammad Anan、Brian Doyle、Emad Ghosheh、Glenn Graessle、Daniel Hein、Prabhaharan Kumarakulasingam、Joe Quint、John Reinke、 Kevin Reynolds、 Sripriya Srinivasan和 Tirumala Vaddiraju。

Kent Back为我提供了许多思路，尤其是他为特殊情况模式所起的名字。Jim Odell将我领入了顾问咨询、教学和写作的世界，我由衷地感谢他。

当我写这本书时，曾将草稿放在Web上。期间许多人通过电子邮件向我指出问题、提出疑问或者讨论其他替代方案。他们中有Michael Banks、Mark Bernstein、Graham Berrisford、Bjorn Beskow、Bryan Boreham、Sean Broadley、Peris Brodsky、Paul Campbell、Chester Chen、John Coakley、Bob Corrick、Pascal Costanza、Andy Czerwonka、Martin Diehl、Daniel Drasin、Juan

Gomez Duaso、Don Dwiggins、Peter Foreman、Russell Freeman、Peter Gassmann、Jason Gorman、Dan Green、Lars Gregori、Rick Hansen、Tobin Harris、Russel Healey、Christian Heller、Richard Henderson、Kyle Hermenean、Carsten Heyl、Akira Hirasawa、Eric Kaun、Kirk Knoernschild、Jesper Ladegaard、Chris Lopez、Paolo Marino、Jeremy Miller、Ivan Mitrovic、Thomas Neumann、Judy Obee、Paolo Parovel、Trevor Pinkney、Tomas Restrepo、Joel Rieder、Matthew Roberts、Stefan Roock、Ken Rosha、Andy Schneider、Alexandre Semenov、Stan Silvert、Geoff Soutter、Volker Termath、Christopher Thames、Volker Turau、Knut Wannheden、Marc Wallace、Stefan Wenig、Brad Wiemerslage、Mark Windholtz、Michael Yoon。

此外，还有许多我不认识或已经遗忘了的人，在此要向他们表达同样真诚的谢意。

最诚挚的谢意依旧要献给我的妻子Cindy，她与我同历风雨，我将永远铭刻在心。

# 目　录

## 第二部分 模 式

# 引 言

构建计算机系统并非易事。随着系统复杂性的增大，构建相应软件的难度将呈指数增大。同其他行业一样，我们只有在不断的学习中进步，从成功经验中学习，从失败教训中学习，才有望克服这些困难。本书中的内容就是这样一些“学习”经验。我希望它们的撰写和编排方式，能够有助于读者更快地学习这些内容，并且，和我在总结出这些模式之前相比，能更有效地与他人进行交流。

在引言中，我想设定本书讨论的范围，并提供一些相关的背景知识与材料。

## 0.1 架构

软件业的人乐于做这样的事——找一些词汇，并把它们引申到大量微妙而又互相矛盾的含义。一个最大的受害者就是“架构”（architecture）这个词。我个人对“架构”的感觉是，它是一个让人印象深刻的词，主要用来表示一些非常重要的东西。当然，我也会小心，不让这些对“系统结构”的“不恭之辞”，影响到读者对本书的兴趣[⊖]。

很多人都试图给“架构”下定义，而这些定义本身却很难统一。能够统一的内容有两点：一点是“最高层次的系统分解”；另一点是“系统中不易改变的决定”。越来越多的人发现：表述一个系统架构的方法不只一种；一个系统中也可能有很多种不同的架构，而且，对于什么在架构上意义重大的看法也会随着系统的生命周期变化。

Ralph Johnson经常在邮件列表上发帖，并提出一些令人关注的见解。就在我完成本书初稿的同时，他又发表了一些关于“架构”的观点。他认为，架构是一种主观上的东西，是专家级项目开发人员对系统设计的一些可共享的理解。一般地，这种可共享的理解表现为系统中主要的组成部分以及这些组成间的交互关系。它还包括一些决定，开发者们希望这些决定能及早做出，因为在开发者看来它们是难以改变的。架构的主观性也来源于此——如果你发现某些决定并不像你想像的那么难以改变，那么它就不再与架构相关。到了最后，架构自然就浓缩成一些重要的东西，不论这些东西是什么。

在本书中，我提出了一些自己的理解，涉及企业应用主要组成部分和我希望能尽早做出的决定。在这些架构模式中，我最欣赏的就是“层次”，将在第1章中进行详细介绍。全书实际上就是关于如何将企业应用组织成不同的层次，以及这些层次之间如何协同工作。大多数重要的企业应用都是按照某种形式的层次分层设计的；当然，在某些情况下，别的设计方式（如管道方式、过滤器方式等）也有它们自己的价值。在本书中我们将不会讨论这些方式，而把注意力

⊖ 因为本书也是关于“架构”的。——译者注